Fast Phong Shading

Gary Bishop
Room 4E-626
David M. Weimer
Room 4F-637
AT&T Bell Laboratories
Crawfords Corner Road
Holmdel, NJ 07733

Abstract

Computer image generation systems often represent curved surfaces as a mesh of planar polygons that are shaded to restore a smooth appearance. Phong shading is a well known algorithm for producing a realistic shading but it has not been used by real-time systems because of the 3 additions, 1 division, and 1 square-root required per pixel for its evaluation. We describe a new formulation for Phong shading that reduces the amount of computation per pixel to only 2 additions for simple Lambertian reflection and 5 additions and 1 memory reference for Phong's complete reflection model. We also show how to extend our method to compute the specular component with the eye at a finite distance from the scene rather than at infinity as is usually assumed. The method can be implemented in hardware for real-time applications or in software to speed image generation for almost any system.

Introduction

Most computer image generation (CIG) systems represent curved surfaces as a mesh of planar polygons because polygons can be transformed and rendered quickly with well known algorithms. Since the polygonal representation is an artifact of the image generation method and is usually of no interest to viewers, (see Figure 1), these systems attempt to restore a smooth appearance to surfaces by varying the intensity across polygons. The efficiency of this shading operation is critical to the performance of a CIG system because it must be performed for one million or more pixels per image. Although algorithms for realistic shading of polygons are well known, real-time CIG systems have not used them because of the large amount of computation they require per pixel. This paper describes a new formulation of Phong shading that drastically reduces the amount of computation compared to previously known formulations. While the new formulation is well suited to implementation with special hardware for real-time display, it is also appropriate for implementation with software for a variety of display systems. Readers who are unfamiliar with surface representation or shading are referred to one of the standard graphics texts for more information, [Newman and Sproull, 1979; Foley and Van Dam, 1983].

Background

Assume, for simplicity, that the input to the shader consists of triangles, in screen coordinates, with normals, $\vec{N}$, to the original curved surface specified at each vertex. The formulations can be extended to polygons with four or more sides.

The shading algorithm must determine the intensity of each point within a triangle from the surface normals, and a vector to the light source, $\vec{L}$. The light source is assumed to be at infinity so that $\vec{L}$ is independent of the point on which the light shines. We will start with diffuse reflection,

$$I_{diffuse} = \frac{\vec{L} \cdot \vec{N}}{|\vec{L}||\vec{N}|},$$

and later show how to extend the results to include Phong's highlight model and an extended highlight model.

Gouraud Shading. The most commonly used shading method in real-time systems, Gouraud shading [Gouraud, 1971], computes the intensity at each point by linear interpolation of the intensities at the vertices. These intensities are determined using the reflection equation above with the normals given at the vertices. The method is popular for real-time systems because it produces images of acceptable quality with only 1 addition per pixel, but the shading has several disturbing characteristics. (1) Moving objects tend to "flatten out" at certain viewing angles, (2) surfaces appear dull or chalky, and (3) images show pronounced Mach bands, exaggerations of intensity change at discontinuities. Figure 2 is a Gouraud shaded chess piece.

Phong Shading. Phong shading, [Phong, 1973], eliminates "flattening out" and dull surfaces and reduces Mach bands but has not, to my knowledge, been used in any real-time system because of the large amount of computation required for its usual formulation. Phong's method determines the intensity of each point using an approximate surface normal that is linearly interpolated from the true surface normals specified at the vertices.

$$\vec{N}(x,y) = \vec{A}x + \vec{B}y + \vec{C}$$

where $\vec{A}$, $\vec{B}$, and $\vec{C}$ are chosen to interpolate the normal across the polygon. Interpolating for successive values of x and y and evaluating the illumination model requires 7 additions, 6 multiplications, 1 division and 1 square root per pixel. Phong proposed a complicated circuit for evaluating this function but it has not, to my knowledge, been implemented. Figure 3 is a Phong shaded chess piece.

Tom Duff has shown, in [Duff, 1979], that Phong shading can implemented more efficiently by combining the interpolation and reflection equations.

$$I_{diffuse}(x,y) = \frac{\vec{L}}{|\vec{L}|} \cdot \frac{\vec{A}x + \vec{B}y + \vec{C}}{|\vec{L}||\vec{A}x + \vec{B}y + \vec{C}|}.$$

Which can be rewritten as

$$I_{diffuse}(x,y) = \frac{\vec{L}\cdot\vec{A}x + \vec{L}\cdot\vec{B}y + \vec{L}\cdot\vec{C}}{|\vec{L}||\vec{L}\cdot\vec{A}x + \vec{L}\cdot\vec{B}y + \vec{L}\cdot\vec{C}|}.$$

Performing the indicated dot products and expanding the vector magnitude yields

$$I_{diffuse}(x,y) = \frac{ax + by + c}{\sqrt{dx^2 + exy + fy^2 + gx + hy + i}}$$

with

$$a = \frac{\vec{L}\cdot\vec{A}}{|\vec{L}|},$$
$$b = \frac{\vec{L}\cdot\vec{B}}{|\vec{L}|},$$
$$c = \frac{\vec{L}\cdot\vec{C}}{|\vec{L}|},$$
$$d = \vec{A}\cdot\vec{A},$$
$$e = 2\vec{A}\cdot\vec{B},$$
$$f = \vec{B}\cdot\vec{B},$$
$$g = 2\vec{A}\cdot\vec{C},$$
$$h = 2\vec{B}\cdot\vec{C}, \text{ and}$$
$$i = \vec{C}\cdot\vec{C}.$$

Using forward differences, this form can be evaluated for successive values of x and y with 3 additions, 1 division and 1 square root per pixel. This is a substantial savings over Phong's formulation but the division and square root are still too expensive for real-time use.

A New Approach

For display purposes we need not evaluate the reflection equation exactly; a good approximation will suffice. The "error" introduced by the approximation will be of no consequence since Phong's normal interpolation and the reflection equation are already approximations. The one-dimensional form of Taylor's series is widely used for approximating functions, such as sine and log, with polynomials. The less widely used, two-dimensional form will serve to approximate the reflection equation.

Taylor's series for a function of two variables is

$$f(a+h,b+k) = f(a,b) + \left(h\frac{\delta}{\delta x} + k\frac{\delta}{\delta y}\right)f(a,b) + \cdots + \frac{1}{n!}\left(h\frac{\delta}{\delta x} + k\frac{\delta}{\delta y}\right)^n f(a,b) + \cdots$$

Shifting the triangle so that $(0,0)$ lies at its center, and expanding in a Taylor series to the second degree produces

$$I_{diffuse}(x,y) = T_5x^2 + T_4xy + T_3y^2 + T_2x + T_1y + T_0$$

with

$$T_5 = \frac{3ig^2 - 4cdi - 4agi}{8i^2\sqrt{i}},$$
$$T_4 = \frac{3cgh - 2cei - 2bgi - 2ahi}{4i^2\sqrt{i}},$$
$$T_3 = \frac{3ch^2 - 4cfi - 4bhi}{8i^2\sqrt{i}},$$
$$T_2 = \frac{2ai - cg}{2i\sqrt{i}},$$
$$T_1 = \frac{2bi - ch}{2i\sqrt{i}}, \text{ and}$$
$$T_0 = \frac{c}{\sqrt{i}}.$$

Using this expansion and forward differences, the intensity can be evaluated with only 2 additions per pixel.

The method can be extended to handle Phong's specular reflection model, $I_{specular} = (\vec{N}\cdot\vec{H})^n$, by using Taylor's series to evaluate the dot product and table lookup to do the exponentiation. (A table of 1k bytes will allow exponentiation to powers up to 20 with less than 1 percent error and a table of 8k bytes allows powers up to 164.) $\vec{H}$ is a vector in the direction of maximum highlight which is halfway between the light direction, $\vec{L}$, and the eye direction, $\vec{E}$.

$$\vec{H} = \frac{\vec{E} + \vec{L}}{|\vec{E} + \vec{L}|}.$$

The eye, like the light source, is assumed for the present to be at infinity. Phong's reflection equation, $I = I_{ambient} + I_{diffuse} + I_{specular}$, can now be evaluated with only 5 additions and 1 memory access per pixel (the ambient component can be rolled into the series for the diffuse component and the series for the specular component can be scaled to allow direct addressing of the exponentiation table). It should be simple to configure hardware to do these operations in less than 100 nanoseconds per pixel.

The additional computation required to determine the T_i for our new method is offset by the greatly reduced computation per pixel for polygons with 10 or more pixels. This estimate is based on the following assumptions: (a) the algorithms are implemented on sequential processors of conventional design, (b) with modern hardware, the time for floating point addition, multiplication, and memory references are all about equal, (c) the computation of $1/\sqrt{x}$ requires about 10 operations, and (d) common subexpressions have been removed from the formulas for T_i. Timings and analysis of the method as implemented in the raster testbed [Whitted and Weimer, 1982] also support a 10 pixel break-even point. The break-even point would be much lower if some simple hardware were added to do the additions and table look-up for our method in parallel. Of course, special hardware could be built for the other methods but it would be much more complicated and probably not as fast.

The images we have produced with this approximation are indistinguishable from those produced with Phong's

method. For polygons with more than about 60 degrees of curvature, this new expansion will produce shadings that are noticeably different from Phong shading but curvatures this large should never be represented by a single polygon.

Going Further

Since the form of the polynomial we have to evaluate at each pixel is independent of the original reflection equation, we can use more elaborate models. One useful extension is to provide for finite eye distance, thus requiring that the vector to the eye, $\vec{E}$, vary across the scene. The resulting variation in the specular component produces more natural looking illumination for scenes rendered with perspective. This is most obvious when there are planar surfaces in the scene because Phong shading with the eye at infinity, like flat shading, renders all parallel surfaces with the same intensity. Figure 4 is a comparison of Phong shading with the eye at infinity and with the eye at the true perspective distance. The reflection equation for the specular component with the eye at a finite distance now becomes

$$I_{specular}(x,y) = \frac{\vec{N}(x,y)\cdot\vec{H}(x,y)}{|\vec{N}(x,y)||\vec{H}(x,y)|},$$

where $\vec{H}(x,y) = \vec{E}(x,y) + \vec{L}$ and $\vec{E}(x,y)$ interpolates the eye vector across the triangle. This can be expanded just as before.

$$I_{specular}(x,y) = \frac{(\vec{A}x + \vec{B}y + \vec{C})\cdot(\vec{D}x + \vec{E}y + \vec{F})}{|(\vec{A}x + \vec{B}y + \vec{C})||(\vec{D}x + \vec{E}y + \vec{F})|}$$

After performing the dot products and expanding the vector magnitude as before.

$$I_{specular}(x,y) = \frac{ax^2 + bxy + cy^2 + dx + ey + f}{\sqrt{(gx^2 + hxy + iy^2 + jx + ky + l)(mx^2 + nxy + oy^2 + px + qy + r)}}$$

with

$$\begin{aligned}
a &= \vec{A}\cdot\vec{D},\\
b &= \vec{A}\cdot\vec{E} + \vec{B}\cdot\vec{D},\\
c &= \vec{B}\cdot\vec{E},\\
d &= \vec{A}\cdot\vec{F} + \vec{C}\cdot\vec{D},\\
e &= \vec{B}\cdot\vec{F} + \vec{C}\cdot\vec{E},\\
f &= \vec{C}\cdot\vec{F},\\
g &= \vec{A}\cdot\vec{A},\\
h &= 2\vec{A}\cdot\vec{B},\\
i &= \vec{B}\cdot\vec{B},\\
j &= 2\vec{A}\cdot\vec{C},\\
k &= 2\vec{B}\cdot\vec{C},\\
l &= \vec{C}\cdot\vec{C},\\
m &= \vec{E}\cdot\vec{E},\\
n &= \vec{D}\cdot\vec{D},\\
o &= 2\vec{D}\cdot\vec{E},\\
p &= 2\vec{D}\cdot\vec{F},\\
q &= 2\vec{E}\cdot\vec{F}, \text{ and}\\
r &= \vec{F}\cdot\vec{F}.
\end{aligned}$$

And expanding in Taylor's series about (0,0).

$$I(x,y) = T_5x^2 + T_4xy + T_3y^2 + T_2x + T_1y + T_0$$

with

$$T_5 = \frac{8al^2r^2 - 4djlr^2 - 4fglr^2 + 3fj^2r^2 - 4dl^2pr + 2fjlpr - 4fl^2mr + 3fl^2p^2}{8l^2r^2\sqrt{lr}},$$

$$T_4 = \frac{4bl^2r^2 - 2dklr^2 - 2ejlr^2 - 2fhlr^2 + 3fjkr^2 - 2dl^2qr}{4l^2r^2\sqrt{lr}} + \frac{-fjlqr - 2el^2pr + fklpr - 2f^2nr + 3fl^2pq}{4l^2r^2\sqrt{lr}},$$

$$T_3 = \frac{8cl^2r^2 - 4eklr^2 - 4filr^2 + 3fk^2r^2 - 4el^2qr + 2fklqr - 4fl^2or + 3fl^2q^2}{8l^2r^2\sqrt{lr}},$$

$$T_2 = \frac{2dlr - fjr - flp}{2lr\sqrt{lr}},$$

$$T_1 = \frac{2elr - fkr - flq}{2lr\sqrt{lr}}, \text{ and}$$

$$T_0 = \frac{f}{\sqrt{lr}}.$$

Summary

We have shown that computer image generation systems can use Phong shading with only a little more computation per pixel than is required for Gouraud shading. This result should allow real-time systems to generate more realistic scenes and conventional systems to produce images more rapidly.

To test the method, we have implemented it as a shader subroutine for the raster testbed. The cpu time for rendering Figure 4 with 14620 polygons at 2048x2048 resolution on a DEC microVAX-II are:

Shading Method	CPU Time	Notes
Gouraud	406s	
Taylor (infinite)	767s	incl 119s for Taylor coeffs
Taylor (finite)	850s	incl 202s for Taylor coeffs
Duff	2597s	incl 1420s in sqrt
Phong	3900s	incl 1405s in sqrt

Acknowledgements

Tom Duff and the reviewers provided many helpful suggestions.

References

Duff, T. 1979. "Smoothly Shaded Renderings of Polyhedral Objects on Raster Displays," *ACM Computer Graphics*, vol. 13, no. 2, pp. 270–275.

Foley, J. D. and A. Van Dam. 1983. *Fundamentals of Interactive Computer Graphics*, Addison Wesley, Reading, MA.

Gouraud, H. June 1971. "Continuous Shading of Curved Surfaces," *IEEE Transactions on Computers*, vol. 20, no. 6, pp. 623–628.

Newman, W. N. and R. F. Sproull. 1979. *Principles of Interactive Computer Graphics*, McGraw-Hill, New York, NY.

Phong, B. T. July 1973. *Illumination for Computer-Generated Images*, Ph.D. Dissertation, Department of Computer Science, University of Utah, Salt Lake City. Gov. ordering no. AD-A008 786.

Whitted, T. and D. Weimer. 1982. "A Software Testbed for the Development of 3D Raster Graphics Systems," *ACM Transactions on Graphics*, vol. 1, no. 1, pp. 43–58.

Figure 1: Flat shading.

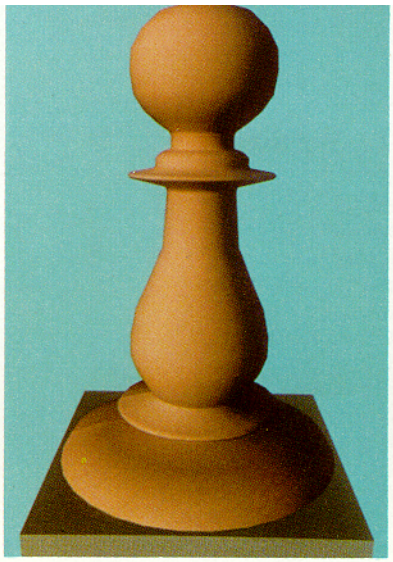

Figure 2: Gouraud shading.

Figure 3: Phong shading.

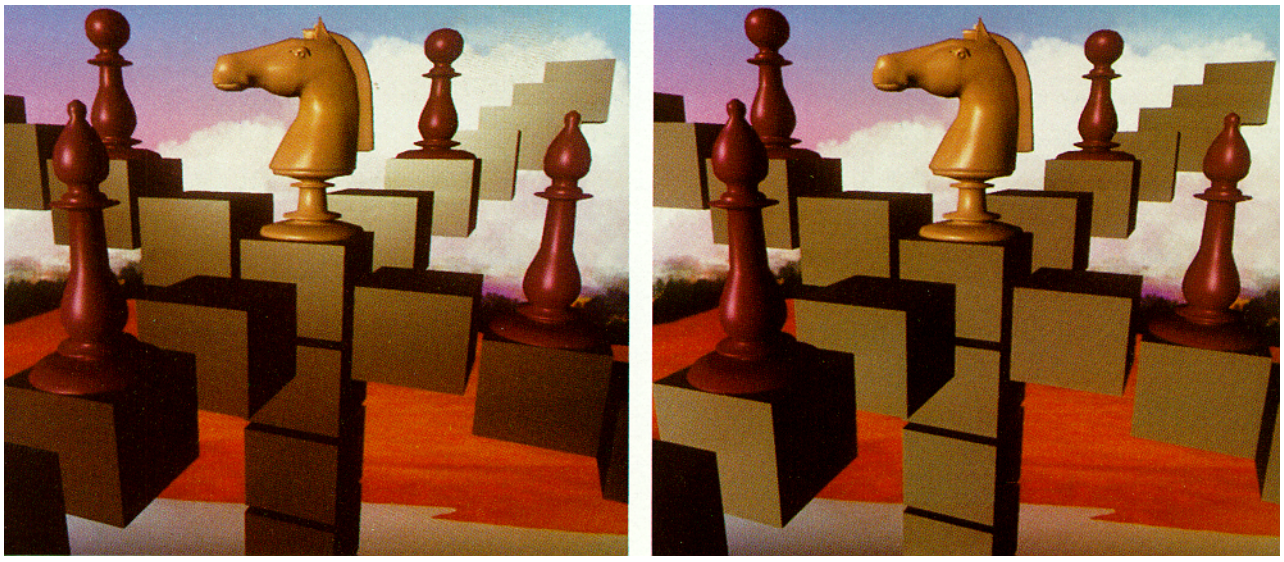

Figure 4: Phong shading with finite (left) and infinite eye distance.